20

REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-01-
0193

1. REPORT DATE *(DD-MM-YYYY)* 1 Dec 1997	**2. REPORT TYPE** Final technical report	[illegible] 1 Aug 1996 - 1 Dec 1997
4. TITLE AND SUBTITLE Combining AI and OR in Heuristics and Optimization		**5a. CONTRACT NUMBER** 31 Jan 98
		5b. GRANT NUMBER F49620-96-1-0413
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) J. N. Hooker		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research / NM, 801 N Randolph St Room 732, Arlington VA 22203-1977		**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR/NM
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
NOTICE OF TRANSMITTAL DTIC. THIS TECHNICAL REPORT
HAS BEEN REVIEWED AND IS APPROVED FOR PUBLIC RELEASE
LAW AFR 190-12. DISTRIBUTION IS UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Mixed logical/linear programming (MLLP) was developed as an extension of mixed integer/linear programming. What appears to be the first practical method of sensitivity analysis for mixed integer/linear programming was developed and applied to a Proctor and Gamble supply chain problem. Consistency-achieving methods of constraint programming were linked with constraint generation methods in operations research. Continuous relaxations were identified for mixed discrete/continuous optimization problems that can accelerate their solution within a logic-based approach. A research monograph, *Logic-Based Methods for Optimization*, was written.

15. SUBJECT TERMS

Optimization, constraint programming, logical inference

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** J. N. Hooker
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		16	**19b. TELEPHONE NUMBER** *(include area code)* 412 268 7589

Combining AI and OR in Heuristics and Optimization: Final Report

AFOSR Grant No. F49620-96-1-0412

J. N. Hooker
Principal Investigator

Graduate School of Industrial Administration
Carnegie Mellon University, Pittsburgh, PA 15213 USA
412-268-7589, fax 412-268-6837, jh38@andrew.cmu.edu

December 1997

1 Objectives

The original objective of this research project was to investigate how ideas from artificial intelligence and operations research can be combined in both optimization methods and heuristic algorithms. The project was proposed by Jon Cagan, Ignatio Grossmann and John Hooker, all of Carnegie Mellon University. The grant award, however, supports only John Hooker. Because his research has recently focused primarily on optimization as opposed to heuristics, the scope of the project has been reduced accordingly.

2 Status of Effort

Substantial progress has been made toward achieveing the objectives of the research. 1) Mixed logical/linear programming (MLLP) was developed as an extension of mixed integer/linear programming. It was shown not only to provide a more flexible modeling framework but in many cases to accelerate solution. 2) What appears to be the first practical method of sensitivity analysis for mixed integer/linear programming was developed and applied to a Proctor and Gamble supply chain problem. It is based on defining the dual of an optimization problem to be a logical inference problem. 3) Consistency-achieving methods of constraint programming were linked with constraint generation methods in operations research. This provides a means of augmenting the cutting plane methods traditionally used in operations research. 4) Continuous relaxations were identified for mixed discrete/continuous optimization problems that can accelerate their solution, within a logic-based approach. In particular, they can drastically speed the solution of some standard location/distribution problems and fixed charge network design problems. 5) A research monograph, *Logic-Based Methods for Optimization,* was brought close to completion.

3 Accomplishments

Accomplishments were obtained in the five areas mentioned above. The first two, MLLP and integer programming sensitivity analysis, represent substantial advances in optimization methods. The third area, which involves consistency-achieving methods, is largely a reinterpretation of previous work. The research on relaxations is in early stages but already points to a substantial improvement in methods for fixed charge network problems.

The research monograph will sum up what is known in its field.

3.1 Mixed Logical/Linear Programming

This represents joint work of J. N. Hooker and M. A. Osorio and is described in [10]. It proposes an extension to mixed integer/linear programming (MILP). It shows how to formulate the discrete aspects of an optimization problem with logical propositions rather than integer variables. It demonstrates with computational tests that MLLP not only has greater modeling flexibility but can substantially accelerate solution of the problem. The paper attempts to draw together previous and new results into a comprehensive description of MLLP. It is based on a rethinking of the role of integer variables. It also provides a framework for integrating the modeling and algorithmic advances of constraint programming with the methods for operations research.

It is likely that future modeling and solution software will combine features associated with mathematical programming (OR) with those associated with constraint programming (AI). This is already beginning to occur and will bring a major change in the way optimization problems are solved. The research described here is designed in part to provide a theoretical and algorithmic basis for this development.

3.1.1 The Idea of MLLP

Mixed logical/linear programming (MLLP), as presented in [10], is a general approach to formulating and solving optimization problems that have both discrete and continuous elements. It extends mixed integer/linear programming (MILP) by introducing logic-based modeling and solution options. MLLP in no way rejects integer programming and in fact incorporates all of its techniques. Its expanded modeling framework may, however, allow more natural or succinct formulations without sacrificing solution efficiency. Its larger repertory of solution techniques may accelerate solution or even solve problems that are intractable for MILP alone. These techniques include branching strategies, relaxations and logic processing algorithms that are not ordinarily associated with integer programming.

Mixed discrete/continuous problems are traditionally conceived as continuous problems in which some of the variables are restricted to be integers. MLLP permits one to take a different view. Rather than embed the discrete aspects of the problem within a linear programming model, which may not be the most natural approach, one can represent them with logical formulas. MLLP therefore has the option of dispensing with integer variables. Rather than require that a feasible solution satisfy a fixed set of inequalities, an MLLP model can contain several alternative sets of inequalities. The logical formulas govern which sets must be satisfied by a feasible solution.

3.1.2 General Form of an MLLP

An MLLP model has the form

$$\begin{array}{ll} \min & cx \\ \text{s.t.} & p_j(y,h) \rightarrow (A^j x \geq a^j),\ \ j \in J \quad \Big| \quad q_i(y,h),\ \ i \in I. \end{array} \tag{1}$$

The constraint set has a logical part (on the right-hand side of the bar) and a continuous part (on the left).

The logical part consists of formulas $q_i(y,h)$ that involve *atomic propositions* $y = (y_1, \ldots, y_n)$, which are either true or false. Such a formula might be $q_1(y,h) = y_1 \vee y_2$, which says that y_1 or y_2 (or both) must be true. There may also be some variables $h = (h_1, \ldots, h_m)$ that take several discrete values. Thus $q_i(y,h)$ could be $(y_1 \vee y_2) \wedge (h_1 \neq h_2)$, where $\wedge$ means 'and.' In general the formulas p_j and q_i may take any form that is convenient for the purpose at hand, provided that their truth value is a function of the truth values of the propositions y and the values of the discrete variables h.

The continuous part associates logical formulas $p_j(y,h)$ with systems $A^j x \geq a^j$ of linear inequalities. A system $A^j x \geq a^j$ is enforced when $p_j(y,h)$ is true. So the constraints of the following problem in effect require x to satisfy $A^1 x \geq a^1$ or $A^2 x \geq a^2$ (or both).

$$\begin{array}{lll|l} \min & cx & & \\ \text{s.t.} & y_1 \rightarrow (A^1 x \geq a^1) & & y_1 \vee y_2 \\ & y_2 \rightarrow (A^2 x \geq a^2) & & \end{array}$$

In general, (x,y,h) is feasible if (y,h) makes all the logical formulas $q_i(y,h)$ true, and x satisfies the linear systems corresponding to the formulas $p_j(y,h)$ that (y,h) makes true.

3.1.3 Solution of an MLLP

The problem (1) can be solved by branching on the truth values of the y_j's and the discrete values of the h_j's. At each node of the search tree, one solves a linear programming problem (LP) containing the constraints that correspond to true p_j's, plus any inequalities added to strengthen the relaxation. A key element of MLLP is to apply a logical inference algorithm to the logical formulas before solving the LP. This may generate *valid constraints* (constraints satisfied by all feasible solutions) in logical form, and in particular it may fix some additional y_j's and h_j's.

An MLLP can therefore be solved in a manner that is analogous to the traditional branch-and-cut algorithms used in MILP. There are two primary differences, however. First, as one descends into the tree, the LP's solved at the nodes are not necessarily defined by fixing certain variables in them. They may also be defined by adding new constraints corresponding to formulas that fixed variables make true, or by some combination of the two methods.

A second difference is that at each node of the search tree, the logical part of the constraint set can be processed with its own set of algorithms, in order to generate additional constraints or check for feasibility. These include many of the logic programming and constraint satisfaction techniques that appear in the computer science and artificial intelligence literatures (discussed below). MLLP therefore provides one means of uniting mathematical programming with methods have been developed more or less independently in other fields.

3.1.4 Motivation for MLLP

The primary rationale for MLLP is that it brings to mathematical programming greater modeling power and a wider range of solution options. But MLLP also grows out of a rethinking of the role of integer variables.

Traditionally integer variables have in most cases served a modeling function and a relaxation function simultaneously. It is proposed here that these functions be separated. When integer variables provide the most natural modeling device for certain constraints, e.g. knapsack constraints, they should be used to formulate those constraints. When a certain portion of the constraint set has a useful continuous relaxation when formulated with integer variables, they should be included in that portion of the problem in order to obtain the relaxation.

In other cases, however, inequalities may not provide the most convenient way to formulate the discrete aspect of the problem. Also their continuous relaxation may be weak, or its effect may be duplicated by adding a few valid inequalities that involve only the original continuous variables. Furthermore, it will be seen that integer variables may have fractional values in the continuous relaxation even when a feasible solution of the original problem has been found. Thus if one branches on integer variables with fractional values, branching may continue unnecessarily.

In such cases, integer modeling may not justify the overhead it incurs. The inclusion of integer variables enlarges the linear programming problems that must be solved at nodes of the search tree. This can be particularly costly when there are many discrete variables, because it may be possible to process the discrete elements of the constraint set much more rapidly in logical form. A simple constraint propagation algorithm, for example, may have the same ability to detect infeasibility in logical constraints as solving the linear relaxation of their inequality formulation. But its speed may be two or three orders of magnitude greater, because it need not carry along the data structures and machinery of a linear solver. Other types of logic processing may obtain valid constraints or fix variables in ways that are not available in MILP.

The primary drawback of MLLP is that it requires more expertise on the part of the user. It provides more options but presupposes that the user knows how to choose the best one. In particular, if integer variables are not used, then the traditional continuous relaxation is unavailable, and it may be necessary to concoct an alternate relaxation.

3.2 Inference-based Sensitivity Aanalysis

This represents joint research with Milind Dawande (who received no support from the grant). Despite the importance of sensitivity analysis in applications, no method of sensitivity analysis for MILP has found acceptance. This paper describes what appears to be the first practical method of MILP sensitivity analysis, at least for moderately-sized problems. It was tested on a supply chain problem involving Proctor and Gamble operations in Mexico. This may be the first time MILP sensitivity analysis has been used in a practical context. The method is based on a general logic-based approach to sensitivity analysis proposed by Hooker in [7]. The key idea is to define the dual of an optimization problem to be a logical inference problem whose solution is a proof. Sensitivity analysis becomes the task of analyzing how the problem data can be changed without invalidating the proof. In MLLP, a sensitivity range for any problem coefficient can be readily found by solving a linear programming problem.

3.2.1 The Basic Idea

A connection between sensitivity analysis and duality has long been recognized. Solution of the linear programming dual, for example, reveals the sensitivity of the optimal value to perturbations in the right-hand sides of the primal constraints. Linear programming duality can be viewed as a special case of *inference duality*, which provides a general approach to sensitivity analysis. In particular, it provides a practical method of sensitivity analysis for integer and mixed integer programming.

Inference duality is based on a fundamental duality of variables and constraints in optimization problems. From one point of view, a given problem concerns what values should be assigned the variables. From the dual point of view, it concerns what may be inferred from the constraints. These two views give rise to two complementary solution approaches: search and inference. The primal problem is solved by search methods that enumerate values of the variables, as in a branching algorithm or heuristic search. The goal is to find optimal values. The dual problem is solved by using inference methods to generate new constraints, as in constraint propagation and cutting plane methods. The goal is to infer a best possible bound on the objective function value. The two approaches can often be profitably combined in such primal-dual methods as branch-and-cut.

Solving the dual problem in a sense *explains* why the optimal value is optimal, because it shows how the optimal value can be derived from the constraints. Sensitivity analysis can be viewed as part of this explanation: it examines the *role of each constraint* in the proof of optimality. It may reveal, for example, that certain constraints play no role at all and can be dropped, or that other constraints can be altered in certain ways without affecting the proof and therefore without making the optimal value worse.

The classical duality-based sensitivity analysis for linear programming has been generalized to nonlinear and integer programming, in the latter case by analyzing how the optimal value depends on the vector of right-hand sides. But the generalization suggested here takes a different direction. Rather than viewing the dual solution as a numerical function of right-hand sides, it views the dual solution as *encoding a proof of optimality.* The classical linear programming dual can be seen as a special case of this, because the dual multipliers specify a linear combination of constraints that prove the optimal value to be optimal. But inference duality can diverge from classical dualities in a more general context.

Unlike other methods of sensitivity analysis, an inference-based method applies at least in principle to any optimization problem. This was established in [7], which introduced the inference-based approach and showed how to apply it to any discrete problem that is solved by a simple branching algorithm.

This approach is adapted in [2] to mixed integer programming (MILP). Existing approaches have never been widely used in practice, due in part to the complexity of computing and interpreting the dual solution. One goal of [2] is to demonstrate that the inference-based approach is computationally feasible and can yield useful information at least in the case of moderate-sized problem instances.

Obviously a key element of the inference-based approach is solving the dual. It was shown in [3] how to obtain a dual solution of a pure 0-1 optimization problem by examining a simple branching tree. The solution takes the form of a proof that uses resolution, a well-known inference technique. This paper generalizes this approach by combining resolution with classical linear programming duality. It shows how to extract a dual solution for any mixed integer/linear problem, even when the branching tree uses linear relaxations, bounds, and certain common classes of cutting planes.

Sensitivity analysis of a minimization problem consists of two parts. One part determines how much the problem can be perturbed without *decreasing* the objective function value more than a specified amount. The other part gives an upper bound on how much the objective function will *increase* if the problem is perturbed by a given amount. The first part of the analysis is the focus of this paper, because it requires solution of the inference dual. The second part of the analysis is straightforward, as it can be obtained by performing classical linear programming sensitivity analysis on the linear relaxation at an optimal leaf node of the search tree.

The two parts of the analysis are asymmetric, but this seems to follow naturally from the asymmetry of primal and dual solutions. One part of the analysis uses a certificate of feasibility for the dual problem (i.e., a proof of optimality), and one uses a certificate for feasibility for the primal problem (an optimal leaf node of the search tree).

3.2.2 Previous Approaches

This analysis may be contrasted with that of Schrage and Wolsey in [11], which is also based on information obtained from the branch-and-bound tree. Their method computes a piecewise linear value function that provides a lower bound on the optimal value that results from perturbing the right-hand sides a given amount. The computation involves the repeated nesting of minima and maxima of linear functionals obtained by solving linear programming duals at nodes of the search tree. Their analysis also provides a bound on the objective function coefficient of a proposed new problem variable, above which the variable will not enter the optimal solution.

The present analysis is more general, in that it permits perturbations of any of the problem data, including the objective function and constraint coefficients as well as the right-hand sides. It also avoids computation of a value function. Rather, it provides a system of linear inequalities, derived from information collected at leaf nodes of the search tree, that is satisfied by any allowable set of perturbations. One can set any desired subset of perturbations to zero and obtain upper and lower bounds on any remaining perturbation by solving two linear programming problems.

Inference duality has other applications. For example, it permits a generalization of Benders decomposition to any optimization problem. Benders cuts are a special case of "nogoods," a well-known idea in the constraint satisfaction literature, but they exploit problem structure in a way that nogoods generally do not. Logic-based Benders decomposition is developed in [3]. Other connections between logical inference and optimization are discussed in [4, 5, 6, 10].

3.2.3 The Inference Dual

Consider a general optimization problem,

$$\begin{aligned} \min \quad & z = f(x) \\ \text{s.t.} \quad & x \in S \\ & x \in D. \end{aligned} \tag{2}$$

The *domain* D is distinguished from the *feasible set* S. For instance, D might be vectors in $\mathcal{R}^n$ or 0-1 vectors. The feasible set S is defined implicitly by a set of constraints that x must satisfy.

To state the inference dual it is necessary to define the notion of implication with respect to a domain D. Let P and Q be two propositions about x; that is, their truth or falsehood is determined by the value of x. *P implies Q with respect to D* (notated $P \xrightarrow{D} Q$) if Q is true for any $x \in D$ for which P is true.

The *inference dual* of (2) is

$$\begin{aligned} \max \quad & z \\ \text{s.t.} \quad & x \in S \xrightarrow{D} f(x) \geq z \end{aligned} \tag{3}$$

So the dual seeks the largest z for which $f(x) \geq z$ can be inferred from the constraint set.

A strong duality theorem is true almost by definition. (Let the optimal value of a minimization problem be respectively ∞ or $-\infty$ when the problem is infeasible or unbounded, and vice-versa for a maximization problem.)

Theorem 1 (Strong Inference Duality) *The optimization problem (2) has the same optimal value as its inference dual (3).*

Proof. If z^* is the optimal value of (2), then clearly $x \in S$ implies $f(x) \geq z^*$, which shows that the optimal value of the dual is at least z^*. The dual cannot have an optimal value larger than z^*, because this would mean that $f(x) = z^*$ cannot be achieved in (2) for any feasible x. If (2) is infeasible, then any z is feasible in (3), which therefore has optimal value ∞. If (2) is unbounded, then (3) is infeasible with optimal value $-\infty$. □

If z^* is the optimal value of the primal problem (2), then solving the dual (3) is tantamount to constructing a proof of $f(x) \geq z^*$ using the constraints as premises. This requires inference rules that are *complete* in a relevant sense; i.e., they provide a way to infer any valid implication of the form $f(x) \geq z$ from the type of constraints that appear in the problem. In the context of linear programming, valid inferences are obtained by taking nonnegative linear combinations of inequality constraints. The completeness of this rule is essentially the content of the classical separation lemmas for polyhedra, as will be seen in the next section.

Note that if a proof of $(x \in S) \xrightarrow{D} f(x) \geq z^*$ is obtained, where z^* is the optimal value, the same proof shows that $(x \in S) \xrightarrow{D} f(x) \geq z^* - \Delta z$ for $\Delta z \geq 0$. This is important in sensitivity analysis, as one often asks what perturbations do not reduce the optimal value below $z^* - \Delta z$ for some given Δz.

3.3 Generating Valid Cuts

The constraint satisfaction literature contains a number of algorithms for achieving "consistency," with the aim of reducing the amount of backtracking necessary to solve a problem by tree search. This paper reinterprets these algorithms as analogous to cutting plane algorithms in integer programming. It is intended to forge a link between constraint satisfaction and integer programming. It introduces mathematical programmers to a concept, namely consistency, that has never been clearly recognized in the field.

3.3.1 Another Perspective on Cuts

Cutting planes have been a major research focus in the operations research community since the 1950's. They make possible the branch-and-cut solution of many integer and mixed integer programming problems that would otherwise be intractable. The rationale for their use almost always relates to the linear relaxation of the problem: they strengthen the relaxation in order to obtain a better bound on the optimal value when the relaxation is solved.

Valid cuts serve another purpose, however, that has not been clearly distinguished in the mathematical programming literature. They can directly reduce the size of the search tree even when they are not used in a relaxation. In fact this perspective gives rise to an alternative theory of valid cuts that in some sense parallels cutting plane theory. Such a theory has been developed in the constraint programming literature. It provides several ways to reduce backtracking that are largely unknown in the mathematical programming community. The paper [8] presents the elementary aspects of this theory in a way that highlights some of the parallels with mathematical programming. It is written for persons with a background in operations research.

3.3.2 An Example

A simple example illustrates how valid cuts can prune a search tree even when they are not used in a relaxation. Suppose that a 0-1 programming problem contains the following constraints.

$$\begin{aligned} x_1 + x_{100} &\geq 1 \\ x_1 - x_{100} &\geq 0 \end{aligned} \tag{4}$$

There is obviously no feasible solution in which $x_1 = 0$. Suppose further that the problem is solved purely by branching on variables in the order $x_1, \ldots, x_{100}$; no relaxations are used. If the $x_1 = 0$ branch is taken first, it may be necessary to search the entire subtree of $2^{100} - 1$ nodes thereby generated in order to discover that it contains no feasible solutions. However, if the valid cut

$$x_1 \geq 1$$

is added to the constraint set, the branch $x_1 = 0$ is ruled out from the start.

A mathematical programmer is likely to view this as an example of preprocessing that fixes a variable. In fact several preprocessing techniques can be viewed as special cases of constraint satisfaction methods. As conceived in operations research, however, preprocessing tends to be a bag of tricks without a unifying theory. The aim here is to provide a theory that not only encompasses many preprocessing ideas but much more as well. The cut $x_1 \geq 1$, for example, can be viewed as making the constraint set (4) "strongly 2-consistent,"[1] which helps to reduce backtracking.

An additional advantage of studying cuts from this perspective is that constraints need not be in inequality form, because they need not occur in linear relaxations. The constraint satisfaction community has taken full advantage of this greater generality. It formulates models with multi-valued discrete variables in addition to boolean and integer variables. It uses logical propositions, all-different constraints, and other non-inequality forms than can give much more succinct formulations of a problem. It may be far from evident how to formulate relaxations in such cases, but the tree-pruning power of valid cuts is still available.

3.4 Continuous Relaxations

Continuous relaxations for discrete problems are normally obtained by writing them a mixed integer/linear programming problems and removing integrality constraints on the variables. Joint research with Hak-Jim Kim has revealed that a much more concise relaxation can sometimes be obtained by projecting the constraints onto the original continuous variables. Projection normally multiplies constraints, but in some case the opposite occurs. In addition, the projected relaxation may have special structure lacked by the traditional one. For example, the relaxations of fixed-charge network flow problems and warehouse location problems become min cost network flow problems when projected. This could enormously acelerate solution of these problems. The technical paper is in preparation.

[1]Technically, to achieve strong 2-consistency, the cut $x_1 \geq 1$ must be used to reduce the domain of x_1 from $\{0, 1\}$ to $\{1\}$ rather than as an explicit constraint. This is because the cut has only one variable.

3.5 Logic-Based Methods for Optimization

A book by this title has been largely completed by J. N. Hooker. It will be a comprehensive treatment of logic-based methods for optimization. Chapters include,

Example of Logic-Based Modeling and Solution
The Logic of Propositions
The Logic of Discrete Variables
The Logic of 0-1 Inequalities
Logic-Based Modeling
Logic-Based Branch and Bound
Constraint Generation
Continuous Relaxations
Branching Heuristics
General Solution Strategies
Inference Duality
Logic-Based Benders Decomposition
Sensitivity Analysis
Relaxation Duality
Discrete Relaxations

4 Personnel Partially Supported

1. *Maria A. Osorio.* At the time of this research Maria Osorio was studying for a Ph.D. under Professor Sergio Fuentes Maya of Universidad Nacional Autónoma de México (UNAM). She visited Carnegie Mellon for a year a half to work with Hooker, during which time she was partially supported by AFOSR. She subsequently received her doctoral degree at UNAM and is on the faculty at Universidad Autónoma de Puebla.

2. *Hak-Jin Kim.* He is a Ph.D. student in operations research at Carnegie Mellon University, supervised by J. N. Hooker. The research described above will become part of his dissertation.

5 Publications (Submitted or Accepted)

The following sre articles related to the research described here that were submitted and/or accepted in the past 12 months.

1. Dawande, M., and J. N. Hooker, Inference-based sensitivity analysis for mixed integer/linear programming (1997), submitted.
2. Hooker, J. N., Constraint satisfaction methods for generating valid cuts, to appear in *Proceedings* of INFORMS Computer Science Technical Section meeting, January 1998.
3. Hooker, J. N., and M. A. Osorio, Mixed logical/linear programming (1997), submitted.

6 Interactions

6.1 Presentations

1. Optimization and constraint satisfaction, tutorial, INFORMS Computer Science Technical Section meeting, Monterey, AC, January 1998 (to be presented by John Hooker)
2. Inference-based sensitivity analysis for MILP, INFORMS Computer Science Technical Section meeting, Monterey, AC, January 1998 (to be presented by John Hooker)
3. Inference-based sensitivity analysis for discrete optimization problems, INFORMS meeting, Dallas, November 1997 (presented by Milind Dawande).
4. Inference-based sensitivity analysis for MILP, International Mathematical Programming Symposium, Lausanne, Switzerland, August 1997 (presented by Milind Dawande).
5. Optimization and constraint satisfaction, Hong Kong University of Science and Technology, July 1997 (presented by J. N. Hooker).
6. Mixed logica/linear programming, Hong Kong Polytechnic University, July 1997 (presented by J. N. Hooker).
7. Optimization and Constraint Satisfaction, 4-hour short course, Escuela Nacional de Optimization y Analysis Numerico (ENOAN'97), Aguascalientes, Mexico, March 1997 (presented by J. N. Hooker).
8. Mixed logical/linear programming, INFORMS meeting, Atlanta, November 1996 (presented by Maria Osorio).

6.2 Industry Interaction

The inference-based method for sensitivity analysis described above was applied to a supply chain management problem in Proctor and Gamble Mexico. The application is described in [1]. The analysis was carried out by the authors of [1] but was not used by the company.

Although mixed integer/linear programming (MILP) has been used for at least two decades, this is to our knowledge the first time that MILP sensitivity analysis has been applied to a life-sized problem from industry.

References

[1] Dawande, M., S. Gavirneni and S. Tayur, Effective heuristics for multi-product shipment models (1997), submitted for publication.

[2] Dawande, M., and J. N. Hooker, Inference-based sensitivity analysis for mixed integer/linear programming (1997), submitted for publication.

[3] Hooker, J. N., Logic-based Benders decomposition (1996). Available on http://www.gsia.cmu.edu/afs/andrew/afs/jh38/jnh.html.

[4] Hooker, J. N., A quantitative approach to logical inference, *Decision Support Systems* **4** (1988) 45-69.

[5] Hooker, J. N., Generalized resolution for 0-1 linear inequalities, *Annals of Mathematics and Artificial Intelligence* **6** (1992) 271-286.

[6] Hooker, J. N., Logic-based methods for optimization, in A. Borning, ed., *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* **874** (1994) 336-349.

[7] Hooker, J. N., Inference duality as a basis for sensitivity analysis, in E. C. Freuder, ed., *Principles and Practice of Constraint Programming–CP96*, Lecture Notes in Computer Science 1118, Springer (1996) 224-236. Also to appear in *Constraints.*

[8] Hooker, J. N., Constraint satisfaction methods for generating valid cuts, to appear in *Proceedings* of INFORMS Computer Science Technical Section meeting, January 1998.

[9] Hooker, J. N. *Logic-Based Methods for Optimization*, in preparation.

[10] Hooker, J. N., and M. A. Osorio, Mixed logical/linear programming (1997), submitted for publication.

[11] Schrage, L., and L. Wolsey, Sensitivity analysis for branch and bound integer programming, *Operations Research* **33** (1985) 1005-1023.